Middlebury at TREC News '21
Exploring Learning to Rank Model Variants

Culton Koster, John Foley
{ckoster, johnf}@middlebury.edu

Department of Computer Science
Middlebury College
Middlebury, VT

Abstract

Middlebury College participated in the TREC News Background Linking task in 2021. We constructed a linear learning to rank model trained on the 2018-2020 data and submitted runs that included variants on the standard low resource learning-to-rank models.

In this notebook paper we detail the contents of our submissions and our lessons learned from this year's participation. We explored a few variant models including a random forest ranker, linear models trained on that random forest, and two-stage linear models, but found that traditional, direct ranking still appears to be optimal.

1 Introduction

The TREC News Task is composed of two core tasks: Background Linking and Wikification. We only participated in Background Linking this year. Given a single news document, the goal is to rank other documents from the rest of the collection with respect to how useful they would be for background reading, in suggestion to a user.

We submitted four runs to the background-linking task, as described in the next section. We describe our indexing and scoring process in Section 2, and our baseline features in Section 3. We discuss why our official runs had such poor performance in Section 4 as well as what our performance is when evaluated on the correct queries. Finally we briefly conclude that our experiments on simple LTR models yielded mostly negative results in Section 5.

1.1 Runs Submitted

`midd-direct` This contains a Coordinate Ascent model trained directly on the baseline LTR features, described in Table 1.

`midd-rf` This contains a Random Forest model trained directly on the baseline LTR features.

`midd-transfer` This a classificaiton model trained on the pseudo-truth output of the random forest classifier.

`midd-linear` This contains the baseline model without the clickbait/spam features.

`midd-twostage` This contains a combination of two linear rankers where the second stage re-ranks only the top-10 documents; creating a nonlinear decision boundary focused on the topmost documents.

In Proceedings of the TREC 2021 Workshop.

2 News-Specific Processing

We first create an index using the `irene` software[1] that is a combination of the `inquery` programming language [2, 12, 3, 7] with the Lucene document indexing system [10].

2.1 Indexing WaPo Articles

We indexed the Washington Post corpus provided for both news and core tasks with the following fields (when available): `published.date`, `url`, `kind`, `title`, `author`, `kicker`, and we constructed a body from the text of (HTML tags were removed with JSoup[2]) the JSON `paragraph` "blobs".

2.2 Result Filtering

Duplicate detection was part of the challenge of working with Washington Post data this year, so we de-duplicated documents by title (choosing the most recent publication date for any conflicts) and rejected all documents without a title from ranking – feeling that documents are not useful for background research if they cannot be summarized concisely.

3 Learning to Rank Baseline

Since there are 211 queries across the four years of TREC News, including this year's test data, there were three years (160 queries) of labeled data. 120 were used randomly as training data, and 40 were used as validation.

One of the most effective and simplest learning to rank models for small datasets is the Coordinate Ascent model [8] because of its ability to directly optimize IR evaluation measures, as our corrected results from this year suggest. As with prior submissions [4], we used the FastRank [5] tool in order to learn these linear models.

3.1 Document Pool Generation

In order to collect candidates for re-ranking, we use a baseline retrieval based on the top-50 terms (excluding inquery[3] stopwords). Each term is weighted based on its frequency in the query document, and a BM25 query with these top-50 terms is generated and submitted against the full collection, with 300 documents collected at first.

We removed any documents that did not have a title field. Following the track guidelines, we also excluded documents whose kickers were: "Opinion(s)", "Letters to the Editor", or "The Post's View".

Finally, we limited our re-ranking pool to the top-200 documents by BM25 score, a feature we kept around as our "pool-score".

3.2 Learning to Rank Features

Our learning to rank features fall into three categories, and are presented in Table 1.

Textual Similarity The first category of features include the standard similarity, the pooling retrieval model, and similarities between the title and document of the query and candidate documents.

Temporal Features Since the Wapo Collection contains a document stream, we incorporated a handful of features using the article's publication date and time, as interpreted by the python `arrow` library.

Quality Features The last category of features we describe are quality & metadata features. They are query-independent and are meant to be a kind of filter on the types of documents that are valuable to recommend to a user [1].

Clickbait Features We also generated a clickbait feature for WaPo articles based on their title, following our approach in prior years [4]. Since we had an extra run, we decided to submit a model with ("midd-direct") and without ("midd-linear") these clickbait features.

[1] `https://github.com/jjfiv/irene`

[2] `https://jsoup.org/`

[3] Sometimes called the Inquery-418 stopword list: `https://github.com/jjfiv/retired-galago/blob/main/src/main/resources/stopwords/inquery`

Group	Feature	Description
Textual Similarity	pool-score	The top-50 words in the query document [6] executed as a weighted-BM25 query [11]
	local-sim	Dot Product of tf-idf vectors as created by "sklearn" [9].
	kicker-sim	We built count-based language models of each kicker tag by accumulating over all documents containing each tag; this is the cosine similarity of the models for the query document and candidate document.
Temporal Models	time-delta	The difference in publication times (seconds).
	day-delta	The difference in publication times (days).
	same-week	True if query and candidate were published within 7 days of each other.
	same-month	True if query and candidate were published within 31 days of each other.
Quality Features	uniq-words	Number of unique words in the document.
	length	Length of candidate document.
	length-ratio	Length of candidate document divided by the length of the query document.
	avg-word-len	A common reading-level approximate; the length of the average word.
	avg-para-len	The average length of the paragraphs in the article (in characters)
Clickbait Features	d-clickbait	Clickbait-probability of candidate title [4]
	q-clickbait	Clickbait-probability of query document title

Table 1: List of Learning to Rank Features

4 Discussion

The official runs from Middlebury manage to get NDCG@5 performance of nearly zero. When we dug into why this happened, it became clear that we had submitted runs that had not found almost any relevant documents.

In early June, the student author on this paper discovered an issue with the officially released queries. The faculty author reported this to track organizers, and a corrected file was released. However, in July, when the faculty author went to regenerate pools for the test data he failed to download the updated query file, thus submitting the wrong queries for each and every query id.

Upon re-running our models on 2021 judgments released with the corrected query files, we're able to explore our differently-trained linear and random-forest models.

Unfortunately, it seems using all features with coordinate ascent directly makes the most sense, and all other experimental settings lead to a loss in quality, despite `midd-rf` and `midd-twostage` theoretically supporting more nonlinear combinations of features.

An interesting finding is that `midd-transfer` performed better on unseen data than `midd-rf`, even though it was trained on the unsupervised output of the `midd-rf` model in an attempt to deal with sparsity of judgment, suggesting that the random forest model we trained was the victim of overfitting.

See Table 2 for full evaluation numbers below.

Submitted Run	Model	Features	Training	NDCG@5	NDCG
`midd-direct`	CA	All	direct	0.452	0.575
`midd-linear`	CA	No Clickbait	direct	0.451	0.560
`midd-rf`	RF	All	direct	0.429	0.557
`midd-transfer`	CA	All	RF	0.449	0.543
`midd-twostage`	CA	No Clickbait	top-10 linear	0.442	0.552

Table 2: Evaluation of submitted models on official qrel-files (with corrected query ids). Official Evaluation metric: NDCG@5

5 Conclusion

When we re-evaluate our results, we find that the exploration of alternative learning-to-rank models mostly led to negative results. Our linear models ("midd-direct" and "midd-linear") broadly outperformed our random-forest model ("midd-rf"). However, our linear model trained on the output of the random-forest model ("midd-transfer") generalized better to the test data than our random-forest model, but it was insignificantly worse than training a

linear model directly. Finally, our two-stage model where we trained an additional linear model to re-rank the top-ten documents led to a small loss.

Acknowledgements

This work was supported in part by Middlebury College. This material is based upon work supported by the National Science Foundation under Grant No. 1827373. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

[1] Bendersky, M., Croft, W. B., and Diao, Y. (2011). Quality-biased ranking of web documents. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 95–104, New York, NY, USA. ACM.

[2] Callan, J. P., Croft, W. B., and Harding, S. M. (1992). The inquery retrieval system. In *Database and expert systems applications*, pages 78–83. Springer.

[3] Cartright, M.-A., Huston, S., and Feild, H. (2012). Galago: A modular distributed processing and retrieval system. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 25–31.

[4] Foley, J., Montoly, A., and Pena, M. (2019). Smith at trec2019: Learning to rank background articles with poetry categories and keyphrase extraction. In *TREC*.

[5] John Foley (2019). FastRank alpha release . `https://jjfoley.me/2019/10/11/fastrank-alpha.html`.

[6] Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM.

[7] Lin, J., Crane, M., Trotman, A., Callan, J., Chattopadhyaya, I., Foley, J., Ingersoll, G., Macdonald, C., and Vigna, S. (2016). Toward reproducible baselines: The open-source IR reproducibility challenge. In *European Conference on Information Retrieval*, pages 408–420. Springer.

[8] Metzler, D. and Croft, W. B. (2007). Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274.

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[10] Pérez-Iglesias, J., Pérez-Agüera, J. R., Fresno, V., and Feinstein, Y. Z. (2009). Integrating the probabilistic models bm25/bm25f into lucene. *arXiv preprint arXiv:0911.5046*.

[11] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

[12] Strohman, T., Metzler, D., Turtle, H., and Croft, W. B. (2005). Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Citeseer.